Using Hadoop: Best Practices

Casey Stella

March 14, 2012

Table of Contents

Introduction

- Hi, I'm Casey
  - I work at Explorys
  - I work with Hadoop and the Hadoop ecosystem daily

Introduction

- Hi, I'm Casey
  - I work at Explorys
  - I work with Hadoop and the Hadoop ecosystem daily
- I'm going to talk about some of the best practices that I've seen
  - Some of these are common knowledge
  - Some of these don't show up until you've been up 'til 3AM debugging a problem.

Introduction

- Hi, I'm Casey
  - I work at Explorys
  - I work with Hadoop and the Hadoop ecosystem daily
- I'm going to talk about some of the best practices that I've seen
  - Some of these are common knowledge
  - Some of these don't show up until you've been up 'til 3AM debugging a problem.
- These are my opinions and not necessarily the opinions of my employer.

The Lay of the Land – The Bad

- There are two APIs, prefer the mapred package
  - The mapreduce and the mapred packages
  - mapred is deprecated, but still preferred
  - Hortonworks just kind of screwed up

The Lay of the Land – The Bad

- There are two APIs, prefer the mapred package
  - The mapreduce and the mapred packages
  - mapred is deprecated, but still preferred
  - Hortonworks just kind of screwed up
- The Pipes interface is really poorly implemented and very slow
- HDFS currently has a single point of failure

The Lay of the Land – The Good

- Hortonworks is actively working on Map-Reduce v2
  - This means other distributed computing models
  - Included in 0.23

The Lay of the Land – The Good

- Hortonworks is actively working on Map-Reduce v2
  - This means other distributed computing models
  - Included in 0.23
- HDFS is dramatically faster in 0.23
  - Socket communication is made more efficient
  - Smarter checksumming

Indexing

- Hadoop is a batch processing system, but you need realtime access

Indexing

- Hadoop is a batch processing system, but you need realtime access
- Options are
  - Roll your own (Jimmy Lin talks about how one might serve up inverted indices in Chapter 3)

Indexing

- Hadoop is a batch processing system, but you need realtime access
- Options are
  - Roll your own (Jimmy Lin talks about how one might serve up inverted indices in Chapter 3)
  - Use an open source indexing infrastructure, like Katta

Indexing

- Hadoop is a batch processing system, but you need realtime access
- Options are
  - Roll your own (Jimmy Lin talks about how one might serve up inverted indices in Chapter 3)
  - Use an open source indexing infrastructure, like Katta
  - Serve them directly from HDFS with an on-disk index aka Hadoop MapFiles

Indexing

- Hadoop is a batch processing system, but you need realtime access
- Options are
  - Roll your own (Jimmy Lin talks about how one might serve up inverted indices in Chapter 3)
  - Use an open source indexing infrastructure, like Katta
  - Serve them directly from HDFS with an on-disk index aka Hadoop MapFiles
  - Serve them through HBase or Cassandra

Indexing

- Hadoop is a batch processing system, but you need realtime access
- Options are
  - Roll your own (Jimmy Lin talks about how one might serve up inverted indices in Chapter 3)
  - Use an open source indexing infrastructure, like Katta
  - Serve them directly from HDFS with an on-disk index aka Hadoop MapFiles
  - Serve them through HBase or Cassandra
  - If data permits, push them to a database

Indexing

- Hadoop is a batch processing system, but you need realtime access
- Options are
  - Roll your own (Jimmy Lin talks about how one might serve up inverted indices in Chapter 3)
  - Use an open source indexing infrastructure, like Katta
  - Serve them directly from HDFS with an on-disk index aka Hadoop MapFiles
  - Serve them through HBase or Cassandra
  - If data permits, push them to a database
- Katta can serve up both Lucene indices and Mapfiles

Indexing

- Hadoop is a batch processing system, but you need realtime access
- Options are
  - Roll your own (Jimmy Lin talks about how one might serve up inverted indices in Chapter 3)
  - Use an open source indexing infrastructure, like Katta
  - Serve them directly from HDFS with an on-disk index aka Hadoop MapFiles
  - Serve them through HBase or Cassandra
  - If data permits, push them to a database
- Katta can serve up both Lucene indices and Mapfiles
- Indexing is hard, be careful.

Performance Considerations

- Setup and teardown costs, so keep the HDFS block size large
- Mappers, Reducers and Combiners have memory constraints
- Transmission costs dearly
  - Use Snappy, LZO, or (soon) LZ4 compression at every phase
  - Serialize your objects tightly (e.g. not using Java Serialization)
  - Key/values emitted from the map phase had better be linear with a **small** constant..preferably below 1

Performance Considerations

- Strategies
  - Intelligent use of the combiners
  - Use Local Aggregation in the mapper to emit a more complex value. (you already know this)
  - Ensure that all components of your keys are necessary in the sorting logic. If any are not, push them into the value.

[1] http://hadoop.apache.org/common/docs/current/mapred_tutorial.html#Profiling

[2] http://hadoop.apache.org/common/docs/current/vaidya.html

Performance Considerations

- Strategies
  - Intelligent use of the combiners
  - Use Local Aggregation in the mapper to emit a more complex value. (you already know this)
  - Ensure that all components of your keys are necessary in the sorting logic. If any are not, push them into the value.
- Profile **JobConf.setProfileEnabled(boolean)** [1]
- Use Hadoop Vaidya[2]

[1] http://hadoop.apache.org/common/docs/current/mapred_tutorial.html#Profiling

[2] http://hadoop.apache.org/common/docs/current/vaidya.html

Unit/Integration Testing Methodologies

- First off, do it.
- Unit test individual mappers, reducers, combiners and partitioners
  - Actual unit tests. This will help debugging, I promise.
  - Design components so that dependencies can be injected via polymorphism when testing

Unit/Integration Testing Methodologies

- First off, do it.
- Unit test individual mappers, reducers, combiners and partitioners
  - Actual unit tests. This will help debugging, I promise.
  - Design components so that dependencies can be injected via polymorphism when testing
- Minimally verify that keys
  - Can be serialized and deserialized
  - *hashcode*() is sensible (Remember: the *hashcode*() for enum is not stable across different JVMs instances)
  - *compareTo*() is reflexive, symmetric and jives with *equals*()
- Integration test via single user mode hadoop

Quality Assurance Testing

- The output of processing large amounts of data is often large
- Verify statistical properties

Quality Assurance Testing

- The output of processing large amounts of data is often large
- Verify statistical properties
  - If statistical tests fit within Map Reduce, then use MR
  - If not, then sample the dataset with MR and verify with R, Python or whatever.
- Do outlier analysis and thresholding based QA

Debugging Methodologies

- Better to catch it at the unit test level
- If you can't, I suggest the following technique
  - Investigatory map reduce job to find the data causing the issue.
  - Single point if you're lucky, if not then a random sample using reservoir sampling
  - Take the data and integrate it into a unit test.

Debugging Methodologies

- Better to catch it at the unit test level
- If you can't, I suggest the following technique
  - Investigatory map reduce job to find the data causing the issue.
  - Single point if you're lucky, if not then a random sample using reservoir sampling
  - Take the data and integrate it into a unit test.
- **DO NOT**
  - Use print statements to debug unless you're sure of the scope.
  - Use counters where the group or name count grows more than a fixed amount.

Debugging Methodologies

- Better to catch it at the unit test level
- If you can't, I suggest the following technique
  - Investigatory map reduce job to find the data causing the issue.
  - Single point if you're lucky, if not then a random sample using reservoir sampling
  - Take the data and integrate it into a unit test.
- **DO NOT**
  - Use print statements to debug unless you're sure of the scope.
  - Use counters where the group or name count grows more than a fixed amount.
- **DO**
  - Use a single counter in the actual job if the job doesn't finish
  - Use a map reduce job that outputs suspect input data into HDFS

Hadoop Opinions

- "We're about a year behind Google" – Doug Cutting, Hadoop World

Hadoop Opinions

- "We're about a year behind Google" – Doug Cutting, Hadoop World
- Giraph and Mahout are just not there yet

Hadoop Opinions

- "We're about a year behind Google" – Doug Cutting, Hadoop World
- Giraph and Mahout are just not there yet
- HBase is getting there (Facebook is dragging HBase into being serious)

Hadoop Opinions

- "We're about a year behind Google" – Doug Cutting, Hadoop World
- Giraph and Mahout are just not there yet
- HBase is getting there (Facebook is dragging HBase into being serious)
- Zookeeper is the real deal

Hadoop Opinions

- "We're about a year behind Google" – Doug Cutting, Hadoop World
- Giraph and Mahout are just not there yet
- HBase is getting there (Facebook is dragging HBase into being serious)
- Zookeeper is the real deal
- Cassandra is cool, but eventual consistency is too hard to seriously consider.

Big Data

- We kind of went overboard w.r.t. Map Reduce
  - Easier than MPI, but really not as flexible.
  - Bringing distributed computing to the masses...meh, maybe the masses don't need it.
  - M.R. v2 opens up a broader horizon

Big Data

- We kind of went overboard w.r.t. Map Reduce
  - Easier than MPI, but really not as flexible.
  - Bringing distributed computing to the masses...meh, maybe the masses don't need it.
  - M.R. v2 opens up a broader horizon
- Data analysis is hard and often requires specialized skills
  - Enter a new breed: the data scientist
  - Stats + Computer Science + Domain knowledge
  - Often not a software engineer

Conclusion

- Thanks for your attention
- Follow me on twitter @*casey_stella*
- Find me at
  - `http://caseystella.com`
  - `https://github.com/cestella`
- P.S. If you dig this stuff, come work with me.